

Figure 1. Search Process, which is applicable to image verification, identification, and retrieval.

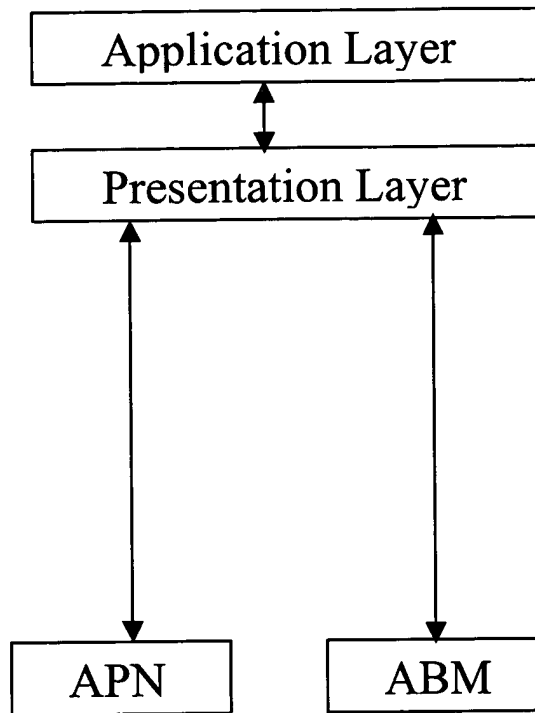


Figure 2. 3-Layer Architecture for software implantation of the Present Invention.

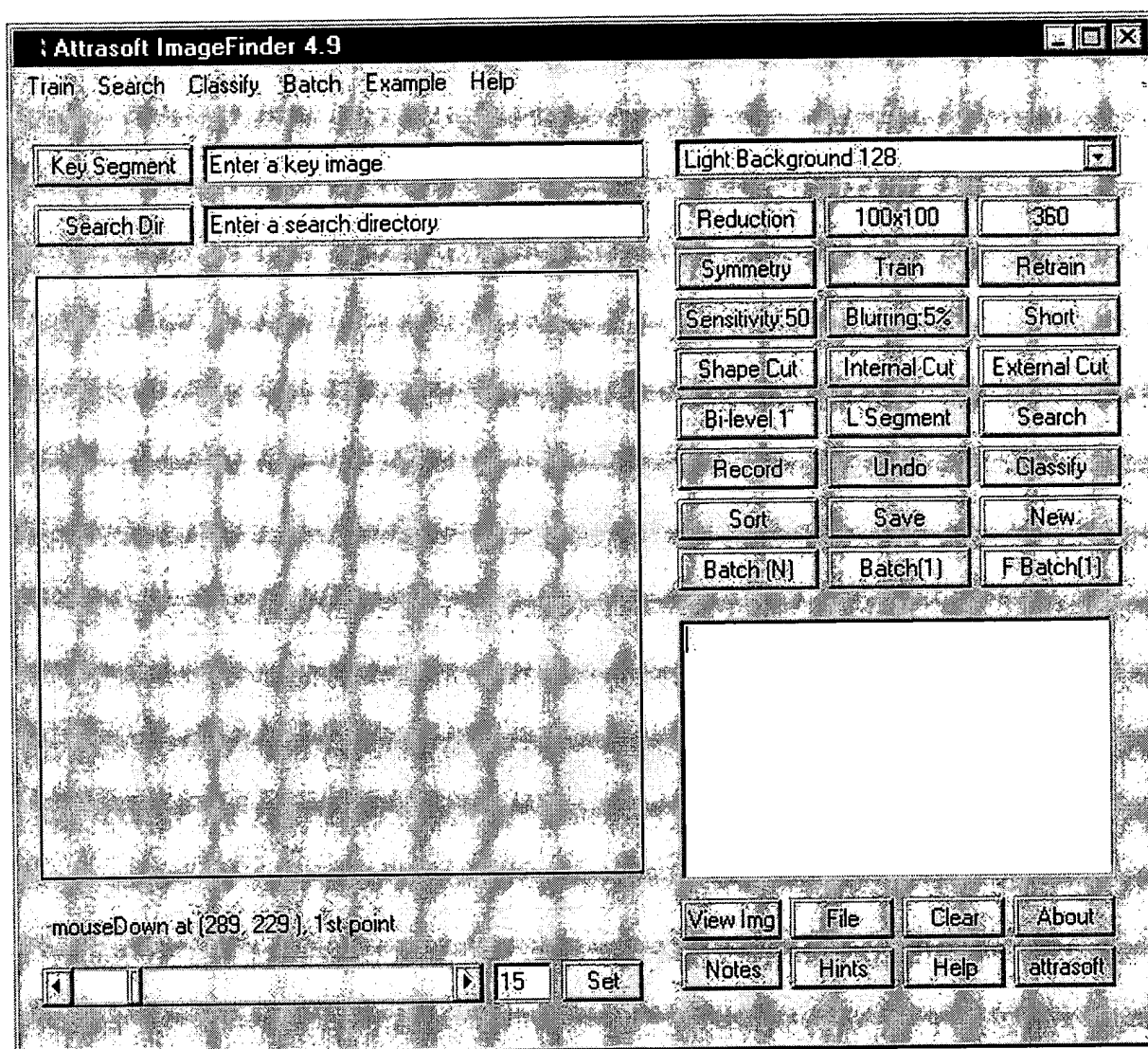


Figure 3. Sample User Interface of the Present Invention.

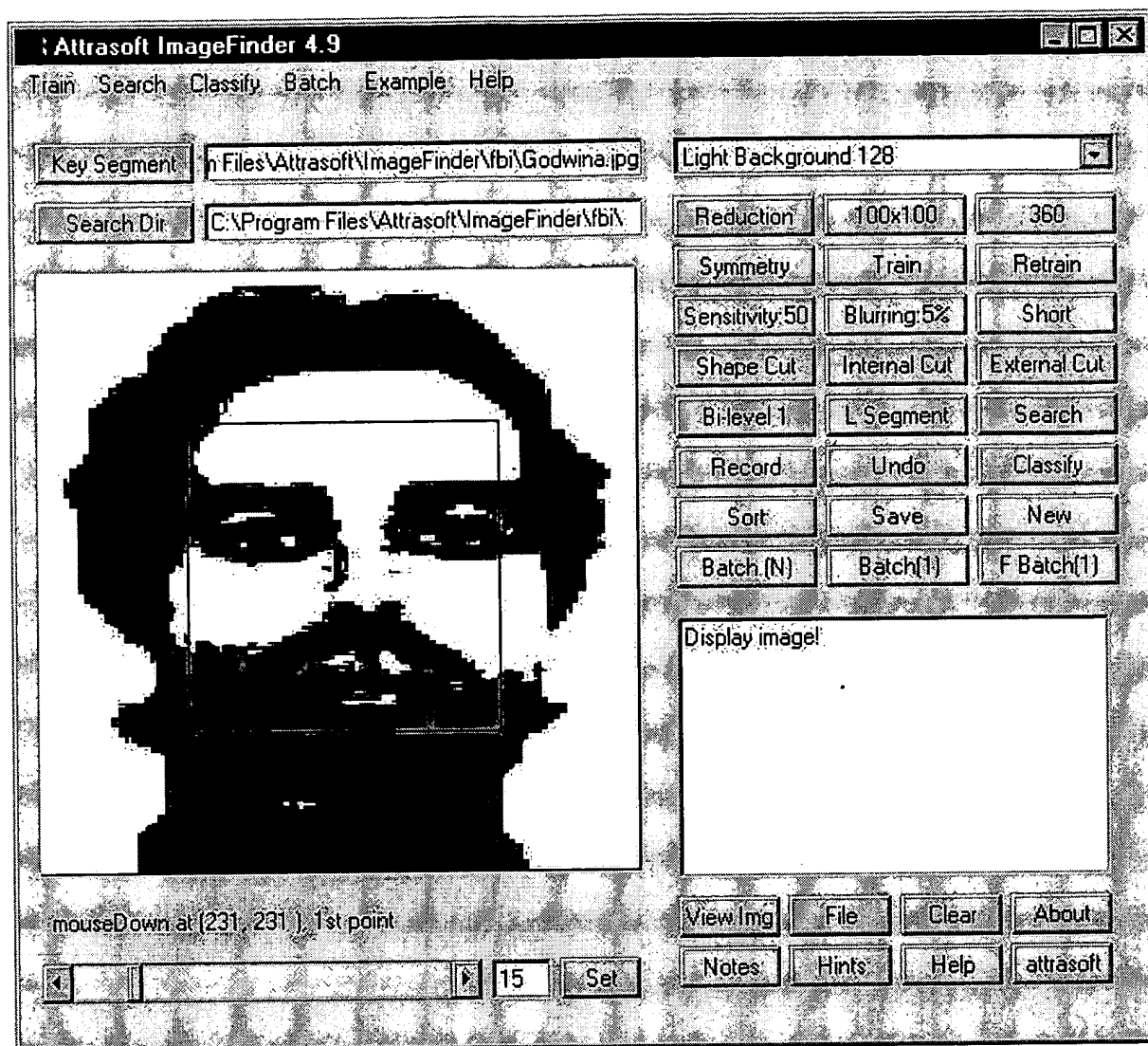


Figure 4. Sample Key Input for the Present Invention.

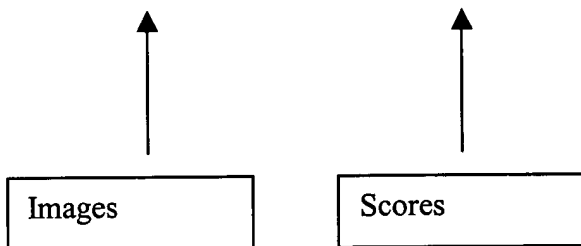
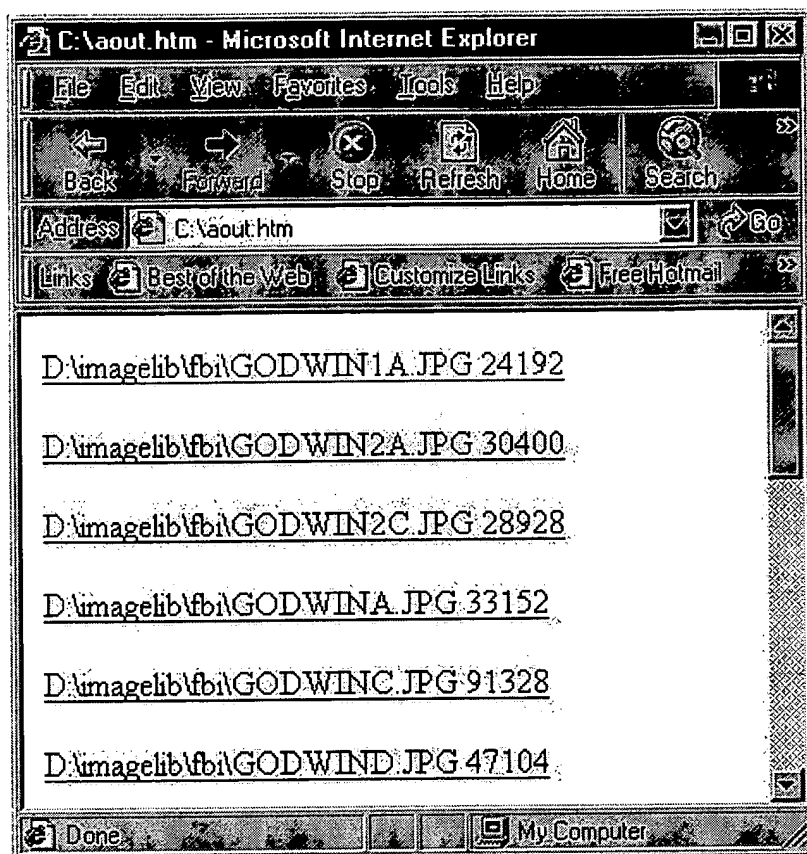


Figure 5. Sample Search Output of the Present Invention. The search output is a list of pairs, or doublets.

For example, line one of the figure 5 shows a double (image, score), where the image is “D:\... GODWIN1A.JPG”, and the score is 24192.

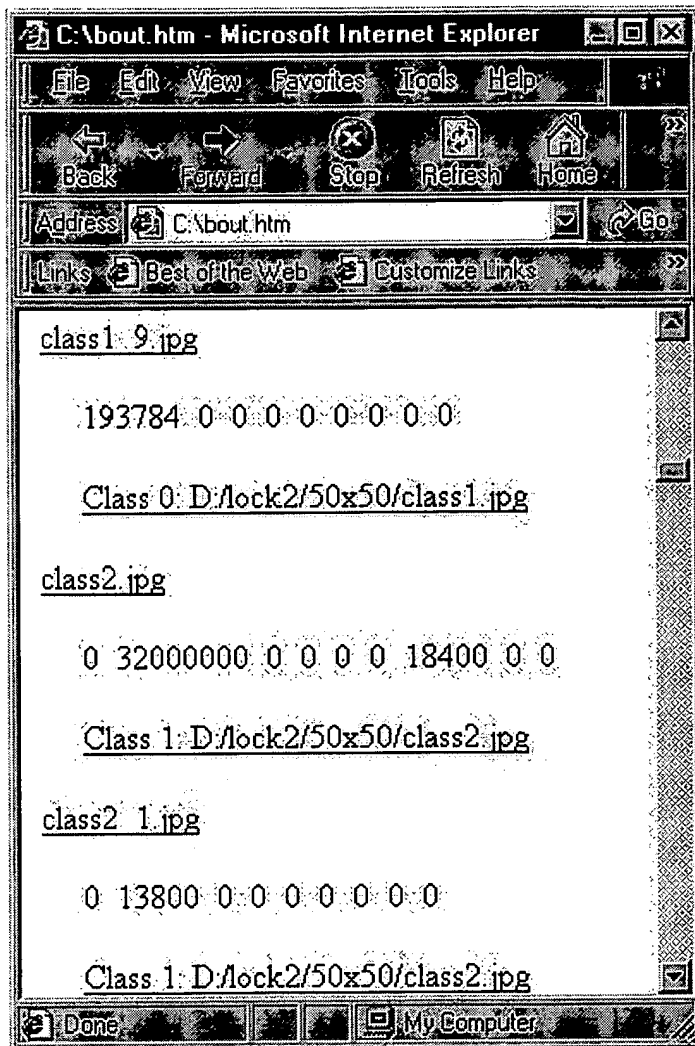


Figure 6. Sample Classification output of the Present Invention. The classification output is a list of triplets.

The N:N classification is implemented by a set of 1:N searches. Let a and b be two different classes. Given an image, x, we can perform a 1:N search for class a, with a triplet (x, a, score-x-a); and a 1:N search for class b, with a triplet (x, b, score-x-b). The final answer is (x, score-x-a, score-x-b; max{score-x-a, score-x-b}). In the above figure, the first line, “class1_9.jpg” is an image, x. The second line has 9 scores, indicating how likely the image x belongs to any of the 9 classes. Finally, Class 0 is the best match because it has the maximum score, 193784. The training image for class 0 is “D:/.../class1.jpg”. The third

line in the above figure shows the classification result of the image, “class1_9.jpg”, which is also a link to the training image, is “D:/.../class1.jpg”.

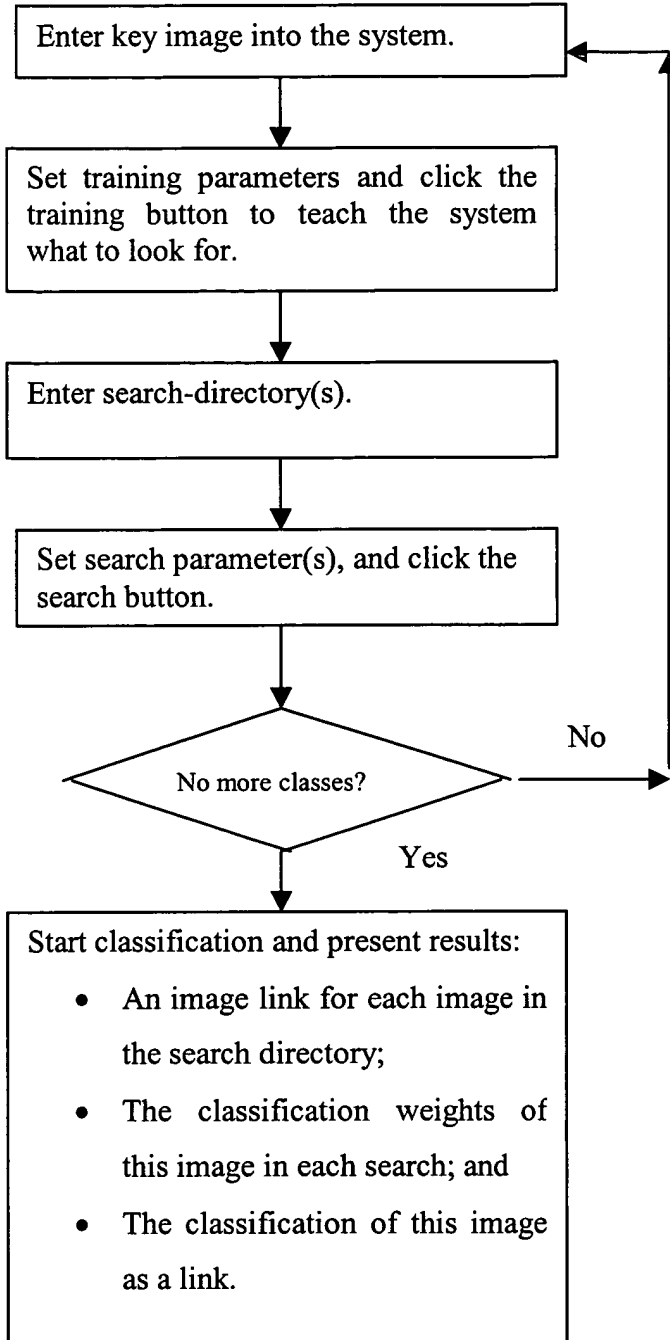


Figure 7. Classification Process, which consists of multiple search processes in Figure 1.

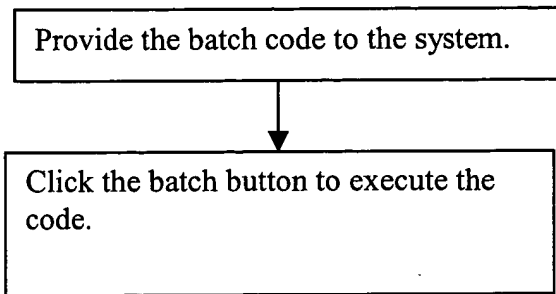


Figure 8. Batch Process, which allows users to duplicate a Search or Classifications in two clicks.

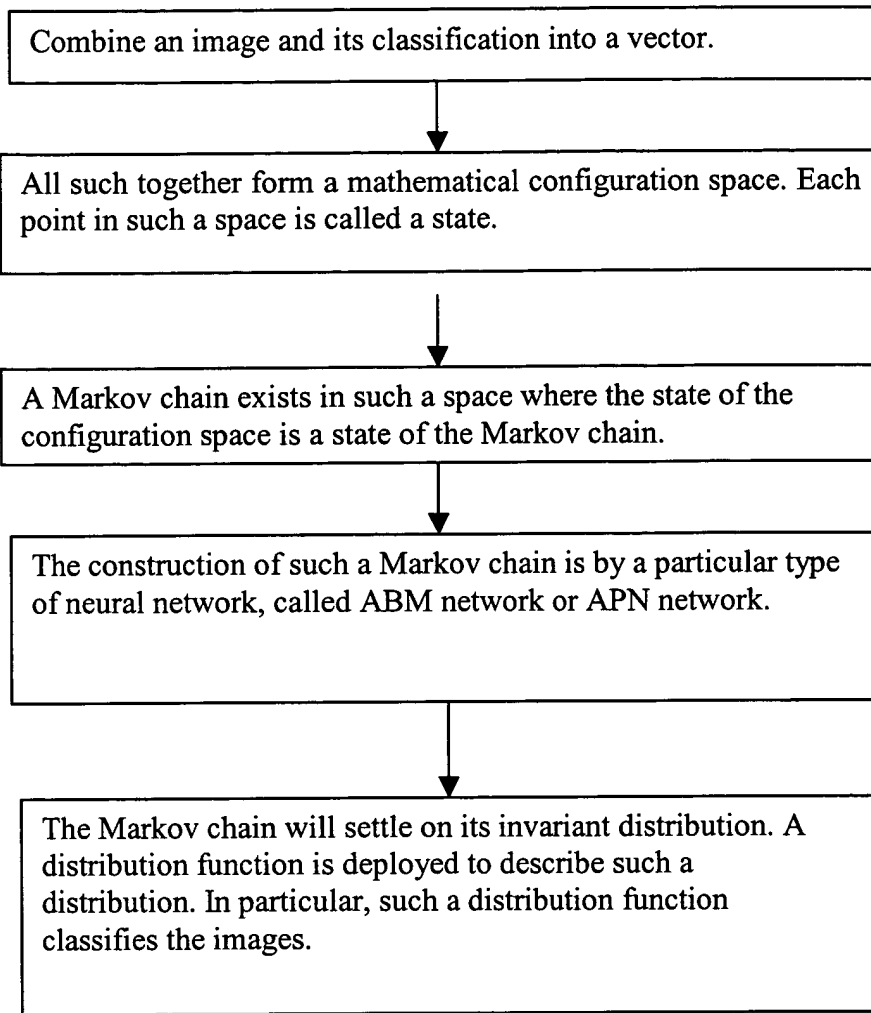


Figure 9. ABM and APN Algorithm Flow Chart.

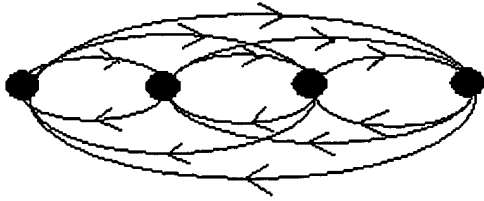


Figure 10. An example of a fully connected artificial neural network with 4 neurons {3, 2, 1, 0}.

| | | | |
|-------|-------|-------|--------|
| p00 | p01 | p02 | p0_15 |
| p10 | p11 | p12 | p1_15 |
| ... | | | |
| p15_0 | p15_1 | p15_2 | p15_15 |

Figure 11. The Markov chain generated by the neural net with 4 neurons in

Figure 10.

The controlling factor in a Markov chain is the transition-probability Matrix. Here p_{ij} is the transition probability matrix element from state i to state j . For example, $p_{1,2}$ is the transition probability from neuron state 0001, where the neuron 0 is excited and the rest are grounded, to neuron state 0010, where the neuron 1 is excited and the rest are grounded. Each neural state is a state in the Markov chain. Assume the neurons are $\{3, 2, 1, 0\}$. The states are $0000 = 0$; $0001 = 1$; $0010 = 2$; ...; $1111 = 15$, i.e. the Markov chain state 0 is a neuron state where all neurons are grounded; the Markov chain state 1 is a neuron state where all neurons are grounded except neuron 0; the Markov chain state 2 is a neuron state where all neurons are grounded except neuron 1; the Markov chain state 3 is a neuron state where neurons 3 and 4 are grounded and neurons 0 and 1 are excited; ...

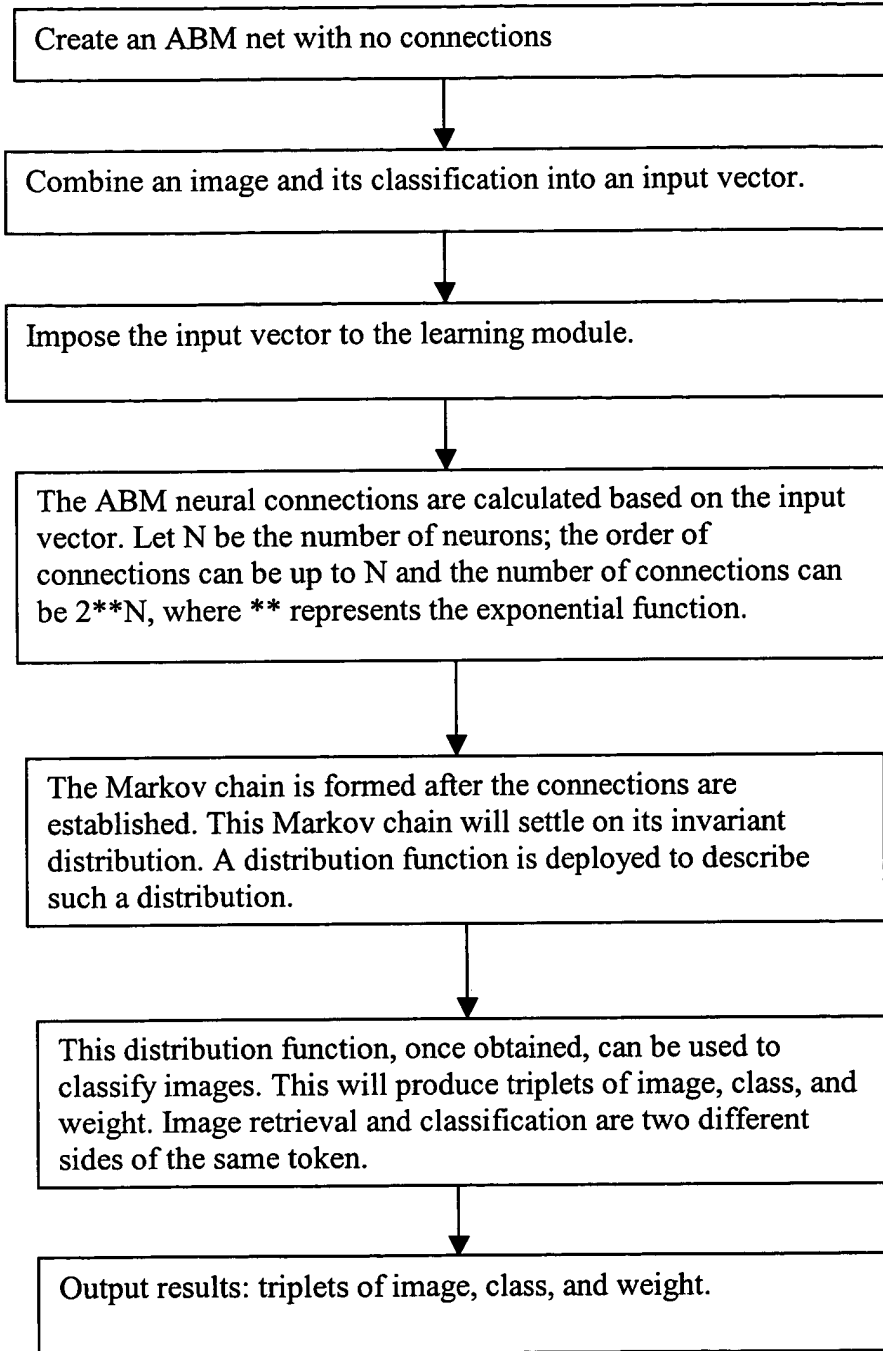


Figure 12. More Detailed ABM Algorithm Flow Chart.

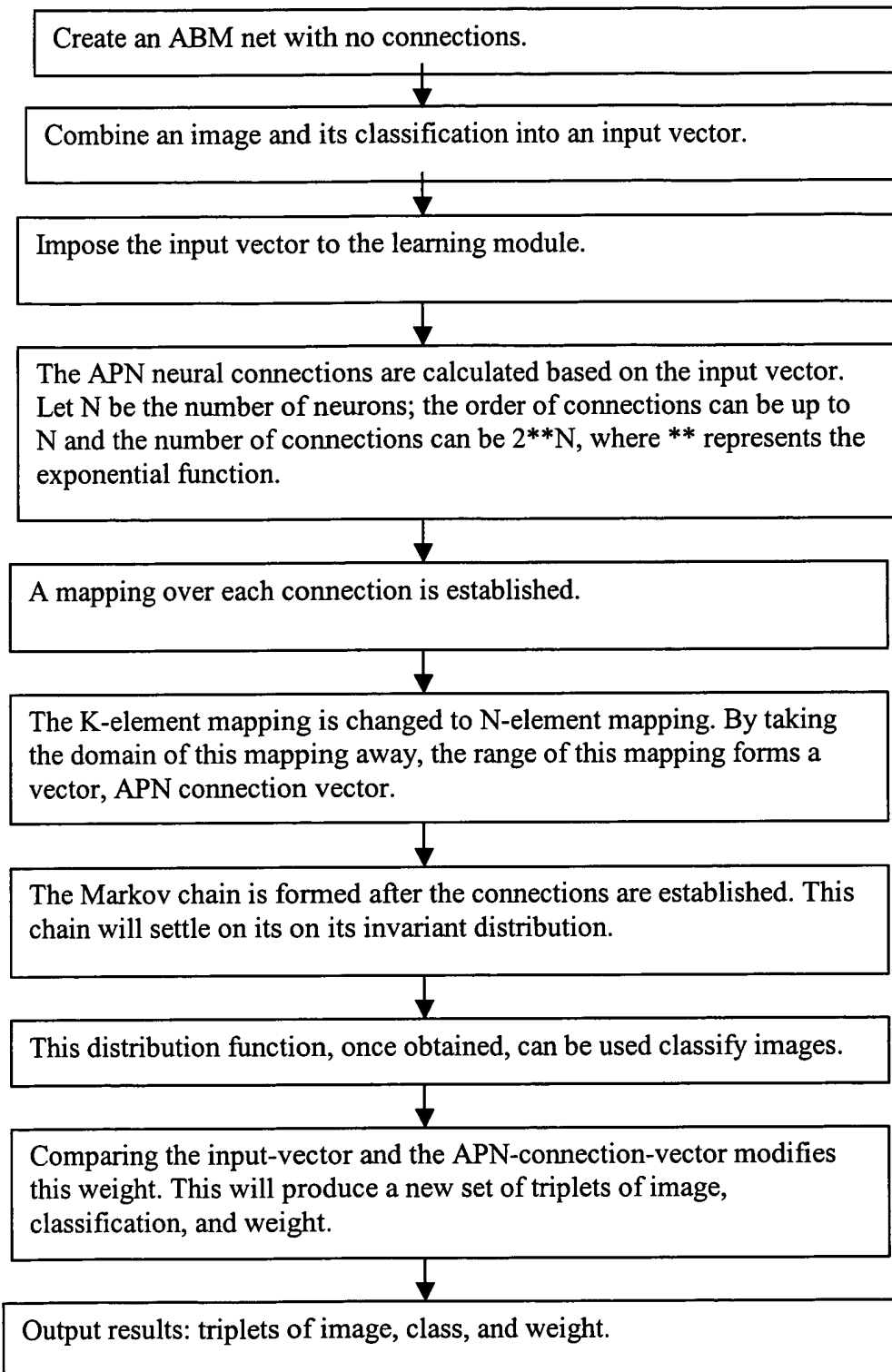
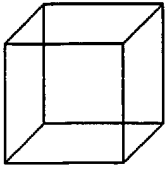


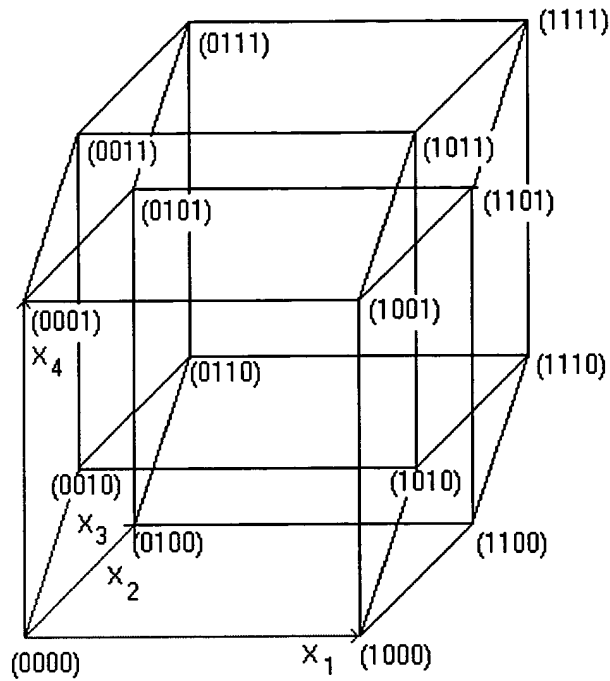
Figure 13. More Detailed APN Algorithm Flow Chart.

The APN algorithm is an extension of the ABM algorithm. The ABM algorithm is for binary images and the APN algorithm is for multi-valued images. The APN algorithm will first use the ABM algorithm to

make an image matching for binary images. If there is no match from the ABM algorithm, the ABM algorithm will output a score 0, which will remain to be 0 when the APN algorithm finishes. If there is a match for the ABM algorithm, then the APN will further modify the ABM results by comparing two input vectors generated by the two comparing images. For example, consider two images (3 5 0 0) and (1 2 0 0), their binary versions are 1100 and 1100. The ABM algorithm will compare the binary images, 0011 and 1100, and return a match. The APN algorithm will take the ABM results, and modify it by the distance between (3 5) and (1 2).

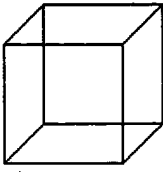


(a) Three-dimensional connection space.

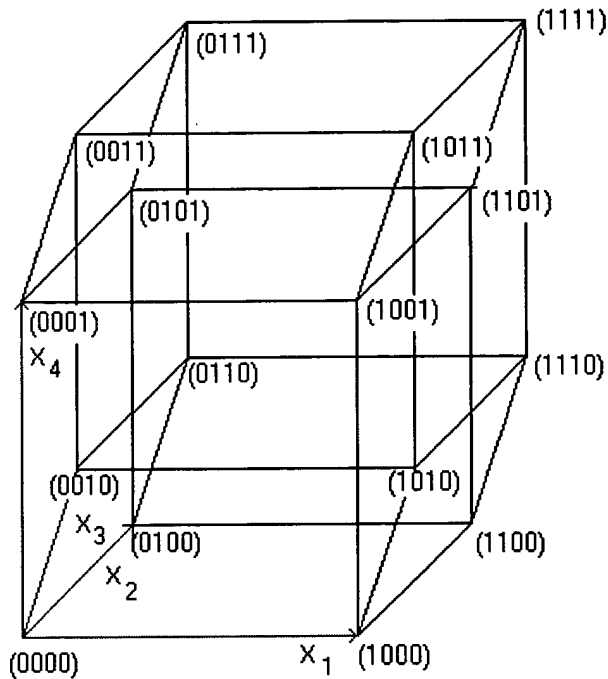


(b) Four-dimensional connection space. Assume two connections are 1100 and 1111, then the Sensitivity distance between them is 2, if the L_1 distance is used, where $L_1 = |x_2 - x_1| + |y_2 - y_1| + \dots$. Here 1100 means that neuron 0 and neuron 1 have a non-zero connection matrix element; and 1111 means there exists a non-zero matrix element between neuron 0, 1, 2, and 3.

Figure 14. Connection Space and the Sensitivity distance.



(a) Three-dimensional binary image space.



(b) Four-dimensional binary image space. Assume a binary image is 1111, we can draw a “sphere” around 1111 with a radius of 1. The following binary images are in this sphere: 1111, 0111, 1011, 1101, 1110. All these images in this “sphere” will be considered the same as the original image, 1111. Blurring is a user-defined parameter that defines this radius. Any image in this sphere is as good as the original image, 1111, for image matching.

Figure 15. Image Space and the Blurring distance.

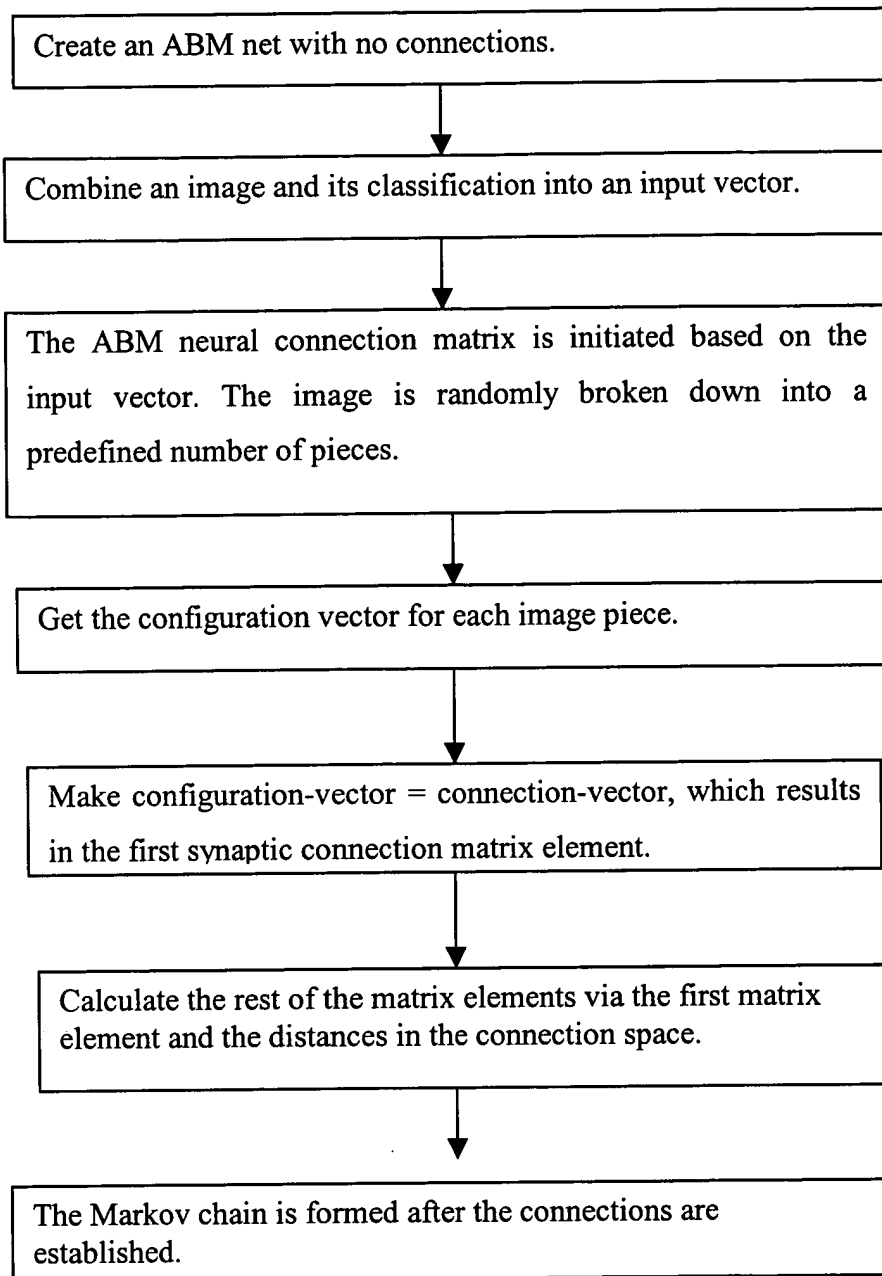


Figure 16. (a) ABM Algorithm.

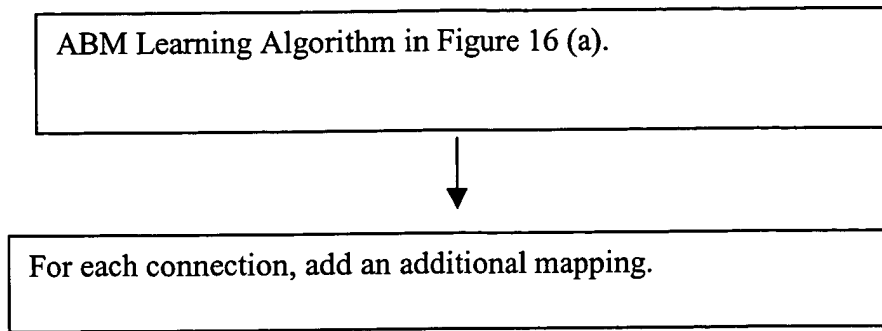


Figure 16. (b) APN algorithm, which is an extension of the ABM algorithm for non-binary values. For example, consider two images (3 5 0 0) and (1 2 0 0), their binary versions are 1100 and 1100. The ABM algorithm will compare the binary images, 0011 and 1100, and return a match. The APN training algorithm will first train the binary ABM net, and then store an additional mapping (0→ 3, 1→ 5) for each connection.

Figure 16. ABM and APN Learning Algorithm Flow Chart.

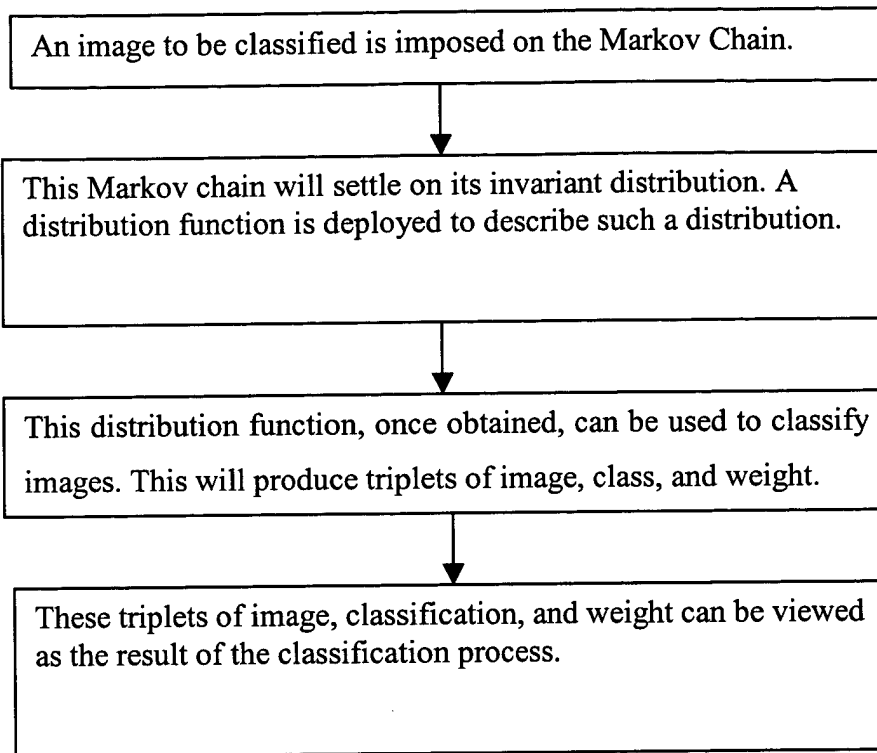


Figure 17. ABM Recognition Algorithm Flow Chart.

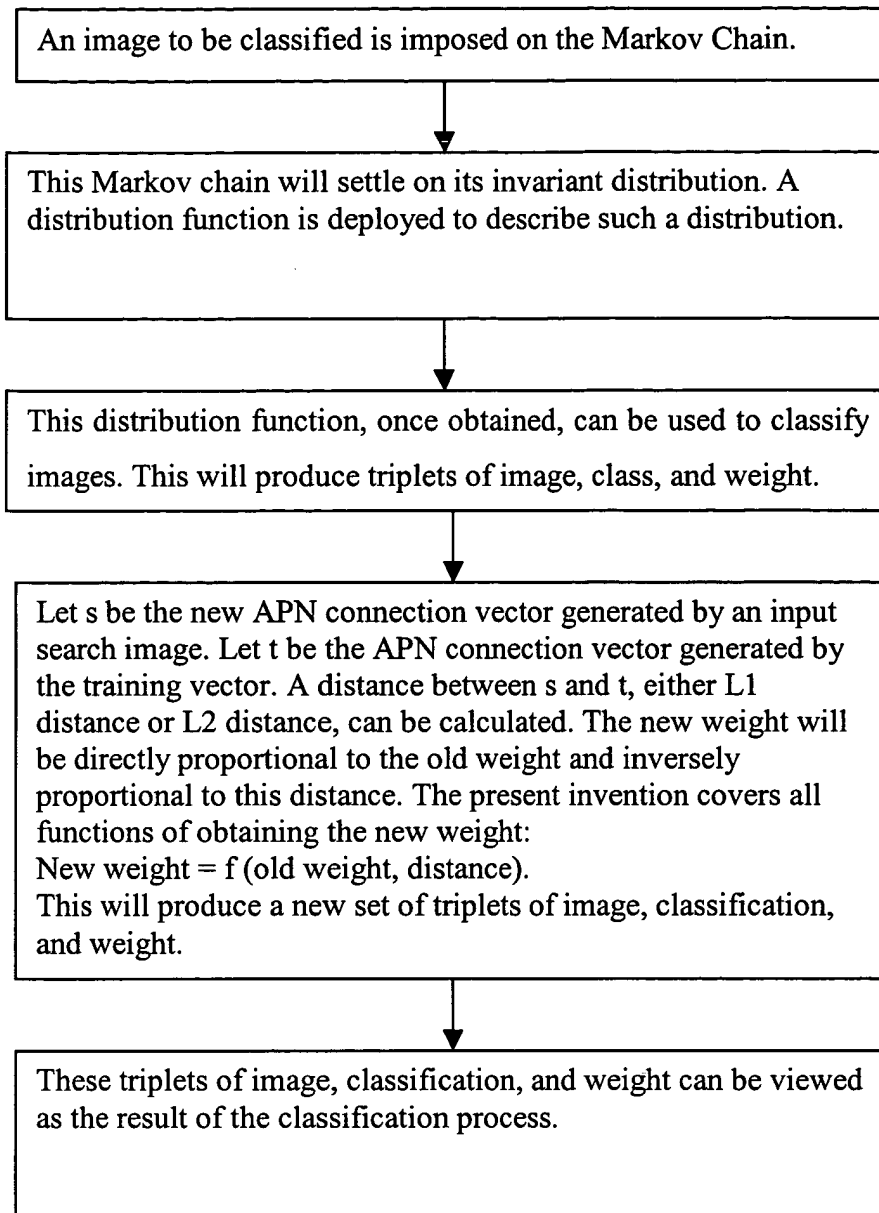


Figure 18. APN Recognition Algorithm Flow Chart.